proceedings

MDPI

Proceedings

Verification of SoC using Advanced Verification Methodology†

Pranuti Pamula[1], Durga Prasad Gorthy[2], Phalguni Singh Ngangbam and Aravindhan Alagarsamy[2,*]

[1] Multi-Core Architecture Computation (MAC) Lab, Department of ECE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, 522501, India; shellypranuti@gmail.com

[2] AMD, Hyderabad,500081; prasad.gorthy@gmail.com

* Correspondence: aravindhan.alagar@gmail.com

† Presented at the Holography Meets Advanced Manufacturing, India, and 15-02-2022.

Abstract: The Semiconductor industry has evolved significantly since its founding in 1950. Primarily used electronic devices such as transistors and diodes but advancements in technology have led to more complex semiconductor devices, from printed circuit boards to multi-million gate design i.e., a System on Chip. Almost 70-80 percent of the total SoC design effort is spent on functional verification. In this paper, verification of interconnect block in a processing system is presented. Trace monitoring of the transactions on the AXI interface of the interconnect is done by programming different operational pointers and filters. Simulated results from the Synopsys - VCS tool.

Keywords: Semiconductor industry; SoC; functional verification; AXI interface

1. Introduction

In recent years, the complexity of an SoC has increased. The more numbers of components in a single chip makes the verification of any SoC design very critical. Hence a proper methodology for any SoC or IP is required [1-4]. Despite all the advancements, there is a significant gap between the modern technology and verification needs for new industries. This situation is getting worse by rapidly changing design as there is rapid movement towards the era of automated vehicles, smart cities and (IoT) Internet of Things [5-8]. Moreover, these electronics devices collect personal information such as location, sleep patterns, health etc., which are stored in the billions of computer devices that operates without pause and even the environment may have compromised or malicious devices. As the system design and security have transformed to adapt themselves so the verification must adjust as well. As per the growing requirements for the design and the time to market, duration has shrunk from years of verification and hard work to less than a year. This aggressive shrinking implies less duration for a thorough design review which may cause misunderstanding in the requirements and a consequent increase in errors. So, the verification is expected to handle more errors in design with even less time duration.

1.1 Problem Statement

Verification of an SoC is carried out on different stages with a different approach as per the design specifications. The interconnect being a common ground for all the rest of the design to interact, there are many functionalities to be verified and connectivity checks to be done. Many tests need to be developed for the verification of an interconnect. Connectivity checks at the interface interconnect being the most important require detailed

Citation: To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Published: date

analysis and thorough research of the design spec. The track sourcing from a master should reach the desired slave interface without any loss in the data packets. Such checks between multiple masters and slave are carried out by initiating write operations to a register space of the slave and expecting to read the same data without any errors. These transactions can be self-tested by the use of System Verilog assertions and checkers. The other functionalities of the design can be verified with a variety of approaches.

2. Materials and Methods

Deep sub-micron effects complicate design closure for very large designs [9]. A system on chip (SoC) is an IC (Integrated Circuit) which is designed by integrating multiple standalone VLSI designs which provide complete functionality for an application. SoC integrates a microprocessor with advanced peripherals such as a coprocessor, memory elements, GPU, Wi-Fi module, etc. This definition of SoC emphasizes the predesigned models of complex design functions which are known as cores. These can be intellectual property blocks, virtual components, macros, etc. In SoC, in-house library cores may be used along with some cores designed by other design houses known as intellectual property. Because of the use of embedded software and increasing integration of cores, the design complexity of SoC has increased dramatically over the past few years. And it's still expected to grow at a very fast rate. According to Moore's law, silicon complexity quadruples every three years [10]. This complexity accounts for the huge size of cores and shrinking geometry.

There are three types of SoCs that are totally distinguishable i.e., an SoC built around a micro-controller, an SoC built around a microprocessor, and a programmable SoC, where the internal elements are not predefined and can be programmed in any manner essential. These kinds of SoCs are also known as FPGAs or a complex programmable logic devices. In all Soc designs, predefined cores are the essential components. The flexibility of the cores depends on the form in which they are available. The trade-off between these cores is in terms of performance, power, speed, area, flexibility, cost, time-to-market, etc. [11].

2.1 Architectural Overview

The SoC architecture integrates a feature of a dual or single-core microprocessor core-based processing system and Xilinx programmable logic in a single device. It is built on state-of-art technology that offers high performance and low power [12]. The multi-core processors are the heart of the PS which also includes on-chip memory, external memory interfaces, and a rich set of I/O peripherals.

SoC offers the flexibility and scalability of an FPG while providing performance, power, and ease of use. The range of devices in the family of SoC enables the designers to find cost sensitive as well as high-performance applications from a single platform using standard tools.

Functional blocks of SoC are shown in Figure 1. The Processing system and Programmable Logic both operate on different power domains. This configuration enables the users to manage the power utilization of PL if required.

The SoC is composed of two major functional blocks:

- Processing System
- Programmable Logic

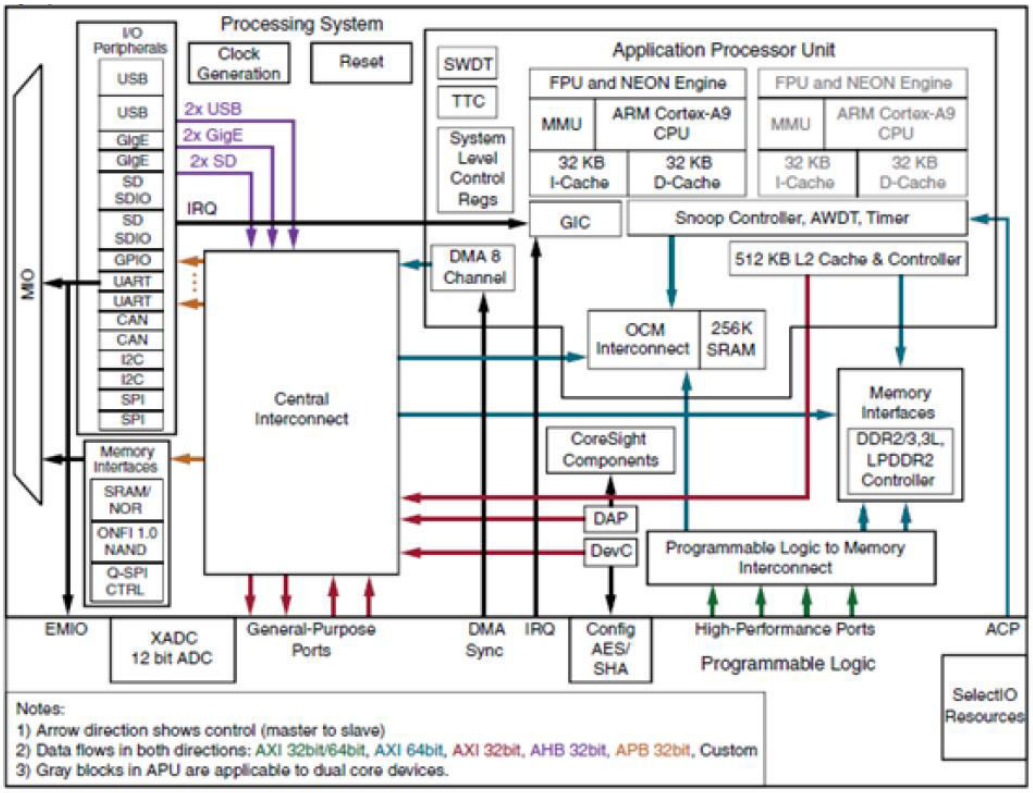

Figure 1. SoC Architecture

2.1.1 Processing System (PS)

- Application Processor Unit: The application processor unit offers high performance and standard-compliant capabilities. The runtime configurations allow the single processor or asymmetrical or symmetrical multiprocessing setups. It is a 32 Kb instruction set with a 32 Kb cache [12]. Also, a sharable 512Kb cache with parity is available. An accelerator coherency port from PL to PS, which is a 64b AXI slave port provides a connection between the processing system and programmable logic. APU also contains a 256Kb of on-chip SRAM which is a dual-ported memory. It is accessible to CPUs, PL, and central interconnects. There are four DMA (Direct memory access) channels for PS, to copy data from CPU memory to/from other system memories.

- Memory interfaces: The memory interface of PS includes multiple memory technologies. It consists of DDR controllers with 16b and 32b widths. This uses up to 73 dedicated pins of PS [12]. The DDR can be powered down as per the idle periods of PS. Transaction scheduling is done for optimizing data bandwidth and latency. The efficiency of memory is increased by 90% by advanced re-ordering engines and 80% efficiency with random read/write operations. A collision check monitors the memory for any write-read collisions and the write buffer is used in that case. The primary boot device can be a NAND controller or parallel SRAM/NOR controller.

- I/O peripherals: The input-output peripherals are a collection of industry-standard interfaces for communication with external systems. The programmable interrupts on the GPIO are used for a status read of raw and masked interrupts. These interrupts are positive edge, negative edge, either edge, high level, or low-level sensitive. A USB 2.0 high-speed on-the-go (OTG) dual-role USB host controllers and USB device controller operations are performed using a single hardware. This configuration uses MIO pins

only. The USB host controller registers and data structures are EHCI compatible [12]. It supports up to 12 endpoints

- Interconnect: The SoC uses several interconnect technologies that are optimized for specific communication requirements of the functional blocks. The SoC interconnect is divided into two parts: one is based on the high-performance data path of AXI on the PS interconnect and the other is on PS-PL interfaces. The PS interconnect consists of an OCM interconnect and a central interconnect. The OCM interconnect provides access to 256Kb memory from the central interconnect and PL. The CPU and ACP interfaces have the lowest latencies to OCM through the SCU. The central interconnect is a 64-bit interconnect. It connects the input-output peripherals and DMA controller to the DDR memory controller, on-chip RAM, and the AXI-GP interfaces for PL logic. It also connects the local DMA units in Ethernet, USB, and SD/SDIO controllers to the central interconnect. It also connects the PS master to the IOP.

2.2 Connectivity check in Interconnect

Interconnect consists of several input and output interfaces. Each of the interfaces reaches out to different slave modules or input from connected masters. The connectivity checks are essential at every interface. This is done to verify the transactions that are intended to pass through a particular interface are reaching without any loss of data packets. Such checks are carried out by initiating write operations from a master to register the space of the slave and expect to read the same data without any errors. This behavior of the design is verified by using system Verilog Assertions and data comparison using system C or System Verilog [13]. The analysis of the simulated result is as important as defining the sequence of the test. The occurrence of an error or unexpected behavior at the output is required to be traced back to the source of the issue. A large amount of time is spent on debugging of simulated results. One of the test scenarios generated to verify connectivity at an interface is as discussed. The simulation result for verification of an AXI interface and APB interface are shown in the waveforms.

3. Results

Many transactions are carried out throughout a design, and with such a complexity level of an SoC, it is required to have better debug features. Trace is the most important debug feature. It is a system that records the execution of the processor and other blocks in real time. This is used for debugging a system that connects and communicates with the external world. Trace all capturing Ture the corner cases that a normal verification of SoC cannot figure out. The trace is used in various ways such as performance optimization of the system, increasing the efficiency of the system and software, and accountability as hard evidence for the cause of system failure. The level of trace monitoring can be flexible and depend on the complexity of the design. In the case of multicore SoC design, the cost of implementation can be reduced. Trace monitoring can be done by using a software or hardware trace. In software trace, data is written to an area of system memory, while a separate task sends this data back to debug block via an available communication channel i.e., JTAG. In hardware trace, a logic watches the address, data, and control signals within the SoC and compresses this information, and sends it to a trace buffer. The buffer subdivides the information into instruction trace, data trace, and bus trace. These signals are then sent to respective debug blocks for further evaluation.

The verification of this block is of utmost importance. This is done by programming a testbench to monitor outgoing traffic with the help of pointers. This outgoing traffic can contain a large bandwidth of data signals. This outgoing data can be filtered as per the requirement and a trace can be generated for only those selected data signals or transac-

tions. The pointers required to monitor the interface are programmed using set of control registers. These configurations are shown in Figure 2.

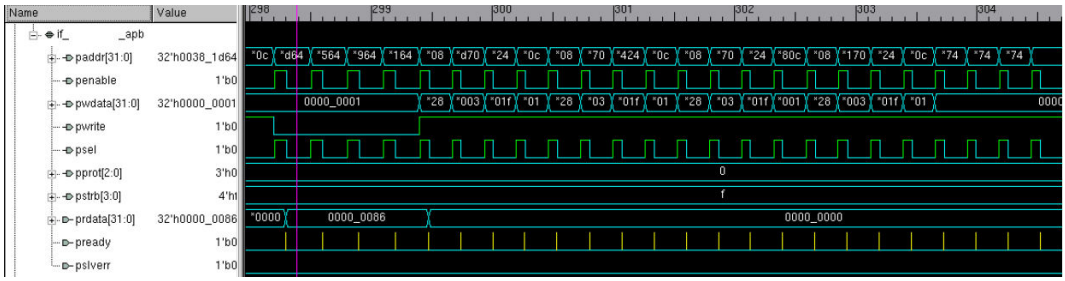

Figure 2: Configuration of Pointers

When pointers are configured and set to monitor a port, the filters are activated to filter out the required amount of data. The filters are configured as per the address or id of the transactions and later subdivided by control/instruction trace, data trace, bus trace, interface trace, fabric trace, etc. Then a burst of AXI transactions is sent to the observed port. A set of such transactions is displayed in the following images.

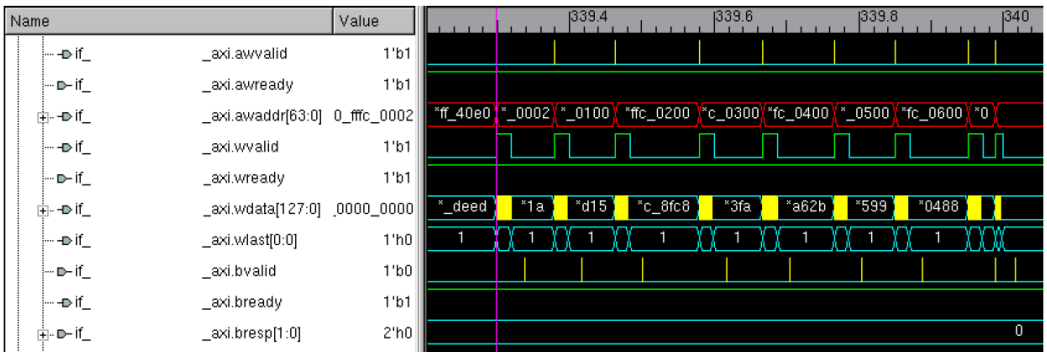

Figure 3: Write operation of AXI Burst Transfer

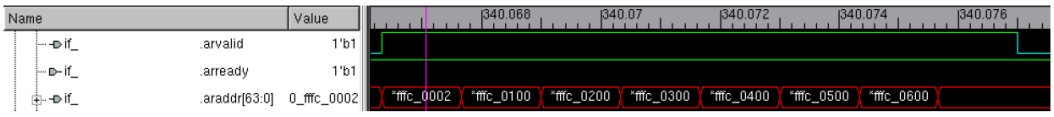

Figure 4: Read operation of AXI Burst Transfer

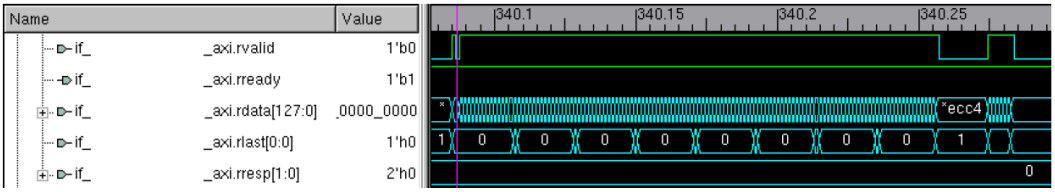

Figure 5: Read Response on AXI Burst Transfer

The write and read burst transfers sent to the AXI bus are shown in Figure 6. The above image shows traffic sent to the observed port. These transactions are then observed with the help of pointers. The transactions on the interface are filtered and sent out to the counter, to count the number of transaction hits. The output is thus observed and analyzed for verification.

Below are the waveforms depicting the counter values at the output. The output of write request pointer is shown in Figure 7.

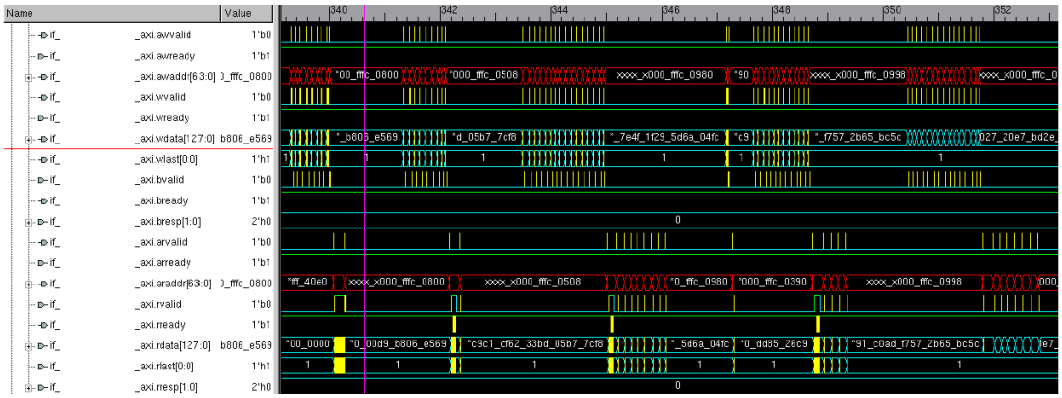

Figure 6: Burst Traffic on AXI Interface

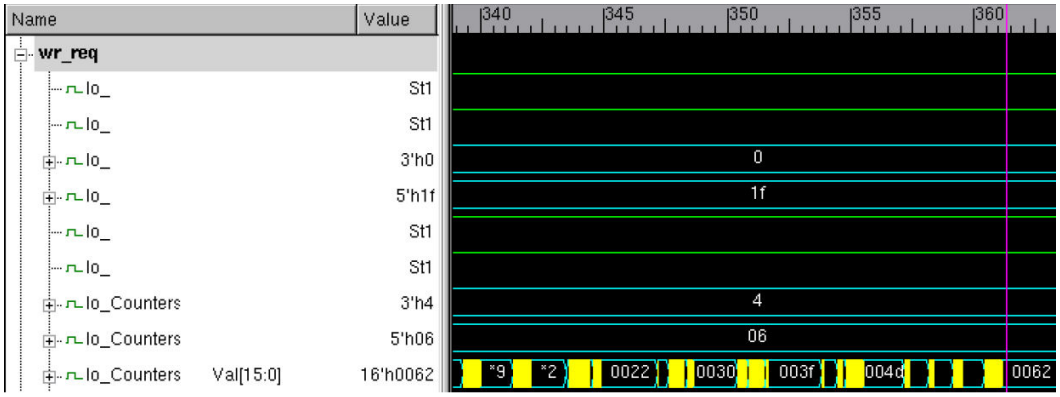

Figure 7: Write Request Transfers

The output of write response pointer is shown in Figure 8.

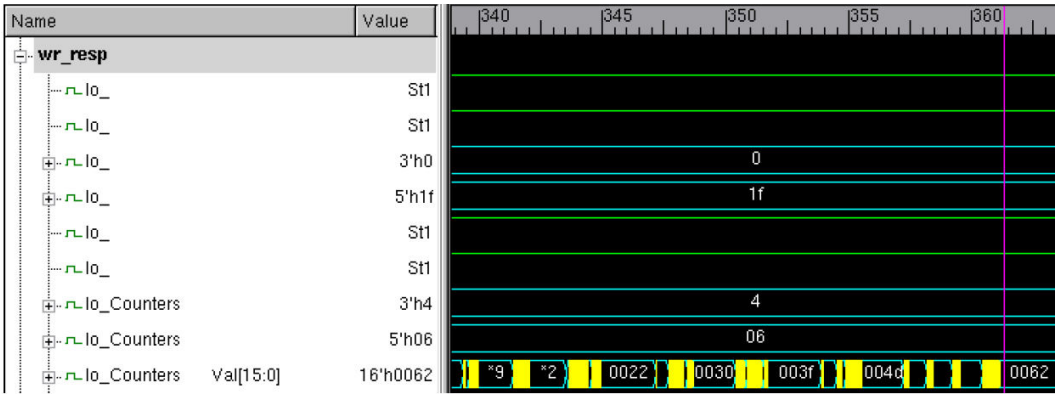

Figure 8: Write Response Transfers

The output of read request pointer is shown in Figure 9.

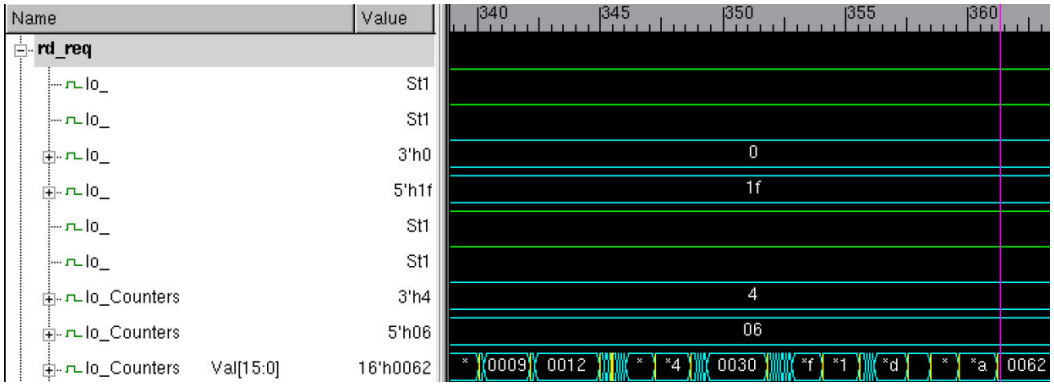

Figure 9: Read Request Transfers

The output of read response pointer is shown in Figure 10**.**

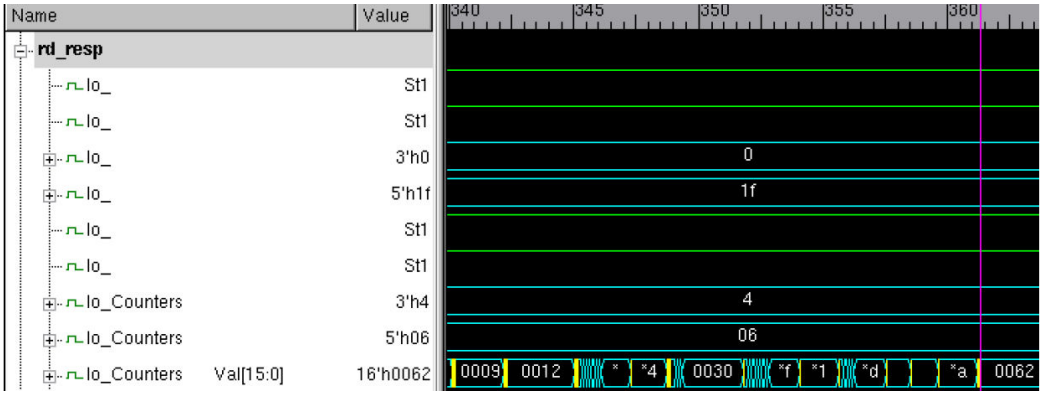

Figure 10: Read Response Transfers

4. CONCLUSIONS.

SoC Verification is a complex and never-ending task. The process can be faster and more efficient when proper programming and simulation tools are used. Verification is done with prior knowledge of SoC architecture and RTL design where the environment is built using UVM and System Verilog. All the parts of the testbench are reusable and can be re-used easily for different designs. This reduces verification complexity and improves efficiency. The design functionalities are verified by using assertions and checkers along with the basic test sequence.

The connectivity of interconnect block with several interfaces is verified successfully. The performance monitoring at various interfaces of interconnect is successfully completed. The simulation results are compared and evaluated by self-checking testbench. This reduces extra efforts to locate the problem or issue in the design. As it locates the exact timestamp and points at the exact line of the RTL code, where a violation has occurred.

5. REFERENCES

[1] P. Ghosh and R. Srivastava, “Case Study: SoC Performance Verification and Static Verification of RTL Parameters,” IEEE Proc 20th International Workshop on Microprocessor/SoC Test, Security and Verification (MTV), pp. 65–72, 2019.

[2] X. Huang, L. Liu, Y. Li, L. Liu and X. Huang, "FPGA Verification Methodology for SiSoC Based SoC Design," IEEE International Conference of Electron Devices and Solid-State Circuits, 2011.

[3] L. Bai, X. Fan, M. Zhang and L. Sun, "A VMM/FPGA Co-verification Method for "Longtium Stream" Processor," IEEE International Conference on Signal Processing, Communication and Computing, 2013.

[4] J. Podivinsky, M. Simkova, O. Cekan and Z. Kotasek, "FPGA Prototyping and Accelerated Verification of ASIPs," 18th IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems, pp. 145-148, 2015.

[5] A. Noami, A. Alahdal, B. P. Kumar, P. Chandrasekhar, N. Safi, "High Speed Data Transactions For Memory Controller Based on AXI4 Interface Protocol SoC", 1st IEEE International Conference on Authorized licensed use limited to: XILINX. Downloaded on February 10,2023 at 13:14:46 UTC from IEEE Xplore. Restrictions apply. Advances in Electrical, Computing, Communications and Sustainable Technologies, 2021.

[6] S. Seongyoung, J. Moon and S. Jun, "FPGA-Accelerated Time Series Mining on Low-Power IoT Devices" Processor," 31st IEEEt International Conference on Application-specific Systems, Architectures and Processors, 2020.

[7] A. Noami, A. A. Alammari, N. Safi, B. P. Kumar, C. C. Paidimarry, "Power Optimization For Multi-Core Memory Controller Using Intelligent Clock Gating Technique", Unpublished

[8] K. C. Gophane and P. C. Bhaskar, "FPGA Based Adaptive IoT Framework for Distinct Applications," in IEEE Fourth International Conference on Computing Communication Control and Automation, 2018.

[9] Najm, Farid, and Jay Abraham. "Accounting for very deep sub-micron effects in silicon models." *EEdesign Magazine* (2001).

[10] Tuomi, Ilkka. "The lives and death of Moore's Law." *First Monday* (2002).

[11] Noguera, Juanjo, and Rosa M. Badia. "System-level power-performance trade-offs in task scheduling for dynamically reconfigurable architectures." Proceedings of the 2003 international conference on Compilers, architecture, and synthesis for embedded *systems*. 2003

[12] Xilinx (2017), 'Zynq-7000 all programmable soc data.

[13] Spear, Chris. System Verilog for verification: a guide to learning the testbench language features. Springer Science & Business Media, 2008.